UNITED STATES PATENT APPLICATION FOR:

SCALABLE DESKTOP

INVENTOR:

RICHARD L. CLARK

ATTORNEY DOCKET NUMBER: NVDA/P001173

CERTIFICATION OF MAILING UNDER 37 C.F.R. 1.10

I hereby certify that this New Application and the documents referred to as enclosed therein are being deposited with the United States Postal Service on March 11, 2004, in an envelope marked as "Express Mail United States Postal Service", Mailing Label No. EV335477757US, addressed to: Commissioner for Patents, Mail Stop PATENT APPLICATION, P.O. Box 1450, Alexandria, VA 22313-1450

Ari Praundji
Signature

Ari Praundji
Name

March 11, 2004
Date of signature

SCALABLE DESKTOP

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to the commonly assigned co-pending United States Patent Application serial number 10/185,764, entitled "METHOD AND APPARATUS FOR DISPLAY IMAGE ADJUSTMENT", filed June 27, 2002 (Attorney Docket No. NVDA/P000551US), which is incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

[0002] Embodiments of the present invention generally relate to a method for displaying an image, and more particularly, to a method for displaying a scalable image.

Description of the Related Art

[0003] Many computer programs provide a feature for zooming in and out of an image. Examples of such computer programs include Adobe Acrobat, MapQuest, Microsoft Word and many others. When a user wants to see a larger version of the image displayed on a computer screen, the user simply selects a scale up or zooming feature. Likewise, when the user wants to reduce the image, the user simply selects a scale down feature. However, this scale up and scale down feature are limited to the specific application or program.

[0004] Therefore, a need exists in the art for a method of scaling up or scaling down an image without being limited to any specific application.

SUMMARY OF THE INVENTION

[0005] Embodiments of the present invention are generally directed to a method for displaying a desktop display surface. The method includes creating a render target surface having substantially the same dimensions as a desktop display

surface, casting the desktop display surface as a texture, and setting the render target surface as a scanout read location.

[0006] In one embodiment, the method further includes creating a two dimensional rectangular object and rendering the two dimensional rectangular object by mapping the desktop display surface texture to the two dimensional rectangular object.

[0007] In another embodiment, the method further includes storing the rendered two dimensional rectangular object to the render target surface and scanning out the rendered two dimensional rectangular object from the render target surface.

[0008] In yet another embodiment, the method further includes receiving a zoom factor, an offset in the x direction and an offset in the y direction; calculating a texture addressing extent configured to determine an amount of the desktop display surface texture to be mapped to the two dimensional rectangular object; and calculating a set of texture addressing offsets in the x and y directions configured to provide the position on the desktop display surface texture from which the desktop display surface texture is to be mapped to the two dimensional rectangular object.

[0009] In still another embodiment, the method further includes calculating the texture addressing coordinates (u, v) as a function of the texture addressing extent and the texture addressing offsets in the x and y directions.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] So that the manner in which the above recited features of the present invention can be understood in detail, a more particular description of the invention, briefly summarized above, may be had by reference to embodiments, some of which are illustrated in the appended drawings. It is to be noted, however, that the appended drawings illustrate only typical embodiments of this invention and are therefore not to be considered limiting of its scope, for the invention may admit to other equally effective embodiments.

[0011] Figure 1 illustrates a simplified block diagram of a computer system 100 according to an embodiment of the present invention.

[0012] Figure 2 illustrates a flow diagram of a method for scaling a desktop display surface in accordance with one embodiment of the invention.

[0013] Figure 3 illustrates a desktop display surface in accordance with one embodiment of the invention.

[0014] Figure 4 illustrates a render target surface having substantially the same dimensions as the desktop display surface illustrated on Figure 3 in accordance with one embodiment of the invention.

[0015] Figure 5 illustrates a set of texture addressing coordinates (u, v) for each corner of the two dimensional rectangular object in accordance with one embodiment of the invention.

[0016] Figure 6 illustrates a rendered two dimensional rectangular object according to one embodiment of the invention.

DETAILED DESCRIPTION

[0017] Figure 1 illustrates a simplified block diagram of a computer system 100 according to an embodiment of the present invention. The computer system 100 includes a central processing unit (CPU) 102 and a system (or main) memory 104 communicating via a bus 106. User input is received from one or more user input devices 108 (e.g., keyboard, mouse) coupled to the bus 106. Visual output is provided on a pixel based display device 110 (e.g., a conventional CRT or LCD based monitor, projector, etc.) operating under control of a graphics processing subsystem 112 coupled to the bus 106. Other components, such as one or more storage devices 128 (e.g., a fixed or removable magnetic disk drive, compact disk (CD) drive, and/or DVD drive), may also be coupled to the system bus 106.

[0018] The graphics processing subsystem 112 includes a graphics processing unit (GPU) 114, a graphics memory 116, and a scanout control logic 120, which may be implemented, e.g., using one or more integrated circuit devices. The graphics memory 116 includes a frame buffer 122 and a texture memory 124. The frame buffer 122 stores pixel data to be read by the scanout control logic 120 and transmitted to the display device 110 for display as an image. In accordance with one embodiment of the invention, the frame buffer 122 includes a desktop display surface 126 and a render target surface 125. A detailed description of the desktop display surface 126 and the render target surface 125 is provided in the paragraphs below with reference to Figures 2-6.

[0019] The texture memory 124 stores data for one or more textures to be used during generation of pixel data. A memory interface 123 is provided to manage communication between the graphics memory 116 and other system components. The memory interface 123 may be integrated with the graphics memory 116 or provided as a separate integrated circuit device.

[0020] The GPU 114 includes various components for receiving and processing graphics system commands received via the bus 106. The GPU 114 may include a front end module 140 and a three-dimensional (3-D) processing pipeline 138 for rendering images, i.e., generating pixel data to be displayed on the display device 110 from 3-D graphics data (e.g., geometry data including polygons and related data describing a scene) received via the bus 106. The GPU 114 may also include a separate two-dimensional (2-D) processing pipeline (not shown) for rendering images using 2-D graphics data received from the CPU 102.

[0021] As mentioned above, the 3-D pipeline 138 is generally used for image rendering. The pipeline 138 may contain various processing modules, such as a geometry processing module 142, a shader 144, a texture blending module 146, and a raster operations module 148, all of which are usable to convert 3-D graphics data into pixel data suitable for displaying on the display device 110. The 3-D pipeline

138 may be controllable by application programs invoking API functions supported by the graphics driver 134 as further described below.

[0022] The computer system 100 further includes a system memory 104, which stores operating system programs 130 for generating pixel and/or graphics data to be processed by the graphics processing subsystem 112. Examples of operating system programs 130 include Graphical Device Interface (GDI) component of the Microsoft Windows operating system. The system memory 104 further stores a graphics driver program 134 for enabling communication with the graphics processing subsystem 112. The graphics driver program 134 implements one or more standard application program interfaces (APIs), such as Open GL, Microsoft DirectX, or D3D for communication with the graphics processing subsystem 112. By invoking appropriate API function calls, the operating system programs 130 are able to instruct the graphics driver program 134 to transfer graphics data or pixel data to the graphics processing subsystem 112 via the system bus 106 and invoke various rendering functions of the GPU 114. Data transfer operations may be performed using conventional DMA (direct memory access) or other operations. The specific commands transmitted to the graphics processing subsystem 112 by the graphics driver 134 in response to an API function call may vary depending on the implementation of the GPU 114, and these commands may include commands implementing additional functionality (e.g., special visual effects) not controlled by the operating system programs 130.

[0023] The system memory 104 further stores various software applications, such as scalable desktop software 132 in accordance with embodiments of the present invention. A detailed description of the operations of the scalable desktop software 132 is provided in the paragraphs below with reference to Figures 2-6.

[0024] It will be appreciated that the computer system 100 is illustrative and that variations and modifications are possible. The computer system 100 may be a desktop computer, server, laptop computer, palm-sized computer, tablet computer, game console, set-top box, personal digital appliance, tethered Internet appliance,

portable gaming system, cellular/mobile telephone, computer based simulator, or the like. The display device 110 can be any pixel-based display, e.g., a CRT or LCD monitor, projector, printer, etc. In some instances, multiple display devices (e.g., an array of projectors or CRT monitors) may be supported, with each device displaying a portion of the image data. The GPU 114 may implement various pipelines for processing 3-D and/or 2-D graphics data, and numerous techniques may be used to support data transfers between the system memory 104 and the graphics memory 116. The GPU 114 or any of its components may be implemented using one or more programmable processors programmed with appropriate software, application specific integrated circuits (ASICs), other integrated circuit technologies, or any combination of these. The graphics memory 116 may be implemented using one or more memory devices. The memory interface 123 may be integrated with the graphics memory 116 and/or the GPU 114, or implemented in one or more separate devices, e.g., ASICs. The scanout control logic 120 may be implemented in the same device (e.g., programmable processor) as the GPU 114 or a different device. In view of the present disclosure, persons of ordinary skill in the art will recognize that the present invention can be embodied in a wide variety of system configurations.

[0025] Figure 2 illustrates a flow diagram of a method 200 for scaling a desktop display surface in accordance with one embodiment of the invention. At step 210, a desktop display surface 126 is created inside the frame buffer 122. The desktop display surface 126 may be a Microsoft Windows desktop display surface. The desktop display surface 126 is generally created by the GPU 114. For purposes of illustrating the invention, a desktop display surface 326 is illustrated on Figure 3. The desktop display surface 326 has dimensions of 32 by 21 pixels. Although the desktop display surface 326 is illustrated as having dimensions of 32 by 21 pixels, various embodiments of the invention described herein are not limited by the dimensions used for purposes of illustrating the invention.

[0026] At step 220, a render target surface 125 is created inside the frame buffer 122. The render target surface 125 may be the same size as, or larger than, the

desktop display surface 126. The render target surface 125 may be created using a DirectX API call. For purposes of illustrating the invention, a render target surface 425 is illustrated on Figure 4. The render target surface 425 has substantially the same dimensions as the desktop display surface 326.

[0027] At step 240, the desktop display surface 126 is cast as a texture. That is, the cast converts the desktop display surface 126 into a texture.

[0028] Once the render target surface 125 is created and the desktop display surface 126 is cast as a texture, then a determination is made as to whether a zoom factor, an offset in the x direction, and an offset in the y direction have been received (step 250). The zoom factor and the offset information may be received through an input device, such as a keyboard, a mouse and the like. The offsets may be in terms of pixels. For purposes of illustrating the invention, the zoom factor is three, the offset in the x direction is four pixels and the offset in the y direction is three pixels.

[0029] If the answer is in the affirmative, then processing continues to step 260 at which a two dimensional rectangular object is created. In creating the two dimensional rectangular object, four or more vertices of the two dimensional rectangular object are determined. In one embodiment, the vertices are positioned on the upper left hand corner, the upper right hand corner, the bottom right hand corner and the bottom left hand corner of the two dimensional rectangular object. In another embodiment, the two dimensional rectangular object has 256 vertices.

[0030] Each vertex generally has five coordinates, i.e., x, y, z, u and v. X and y generally refer to the location of that vertex with respect to the x and y dimensions of the display area. For purposes of illustrating the invention, for a 32 by 21 display area, the (x, y) coordinates for the upper left hand corner vertex are (0, 0). The (x, y) coordinates for the upper right hand corner vertex are (32, 0). The (x, y) coordinates for the lower left hand corner vertex are (0, 21). The (x, y) coordinates for the lower right hand corner vertex are (32, 21). Z refers to the depth coordinate of a vertex. For a two dimensional object, z is set to a constant value. U and v refer

to the texture addressing coordinates, which are typically normalized to be in the range from 0 to 1.

[0031] The texture addressing coordinates (u, v) are configured to control how the desktop display surface texture is to be mapped to the two dimensional rectangular object. The texture addressing coordinates (u, v) are a function of a texture addressing extent, a texture addressing offset in the x direction, and a texture addressing offset in the y direction. The texture addressing extent provides the amount of the desktop display texture to be mapped to the two dimensional rectangular object. The texture addressing extent is calculated as the texture address range divided by the zoom factor. In a case of texture addressing coordinates normalized to a texture address range of 0 to 1, the texture addressing extent is equal to (1-0)/zoom factor (or the inverse of the zoom factor). For purposes of illustration, since the zoom factor is three, the texture addressing extent is 0.333.

[0032] The texture addressing offsets in the x and y directions provide the position on the desktop display texture from which the desktop display texture is to be mapped to the two dimensional rectangular object. The texture addressing offset in the x direction is calculated as the offset in the x direction (received at step 250) divided by the number of pixels in the x direction of the display area. The texture addressing offset in the y direction is calculated as the offset in the y direction (received at step 250) divided by the number of pixels in the y direction of the display area. For purposes of illustration, since the offset in the x direction is four pixels and the offset in the y direction is three pixels, then the texture addressing offset in the x direction is 4/32 (or 0.125) and the texture addressing offset in the y direction is 3/21 (or 0.143) for a 32 by 21 display area.

[0033] Once the texture addressing extent and the texture addressing offsets are determined, then the texture addressing coordinate u for the upper left hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the x direction, while the texture addressing coordinate v for the upper left

hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the y direction. Following along the illustration given above, the texture addressing coordinates (u, v) for the upper left hand corner of the two dimensional rectangular object is (0.125, 0.143).

[0034] The texture addressing coordinate u for the upper right hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the x direction plus the texture addressing extent, while the texture addressing coordinate v for the upper right hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the y direction. Following along the illustration given above, the texture addressing coordinate u for the upper right hand corner of the two dimensional rectangular object is 0.125 plus 0.333, which is equal to 0.458. The texture addressing coordinate v for the upper right hand corner of the two dimensional rectangular object is 0.143. Thus, the texture addressing coordinates (u, v) for the upper right hand corner of the two dimensional rectangular object is (0.458, 0.143).

[0035] The texture addressing coordinate u for the bottom left hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the x direction, while the texture addressing coordinate v for the bottom left hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the y direction plus the texture addressing extent. Following along the illustration given above, the texture addressing coordinate u for the bottom left hand corner of the two dimensional rectangular object is 0.125. The texture addressing coordinate v for the bottom left hand corner of the two dimensional rectangular object is 0.143 plus 0.333, which is equal to 0.476. Thus, the texture addressing coordinates (u, v) for the bottom left hand corner of the two dimensional rectangular object is (0.125, 0.476).

[0036] The texture addressing coordinate u for the bottom right hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the x direction plus the texture addressing extent, while the texture

addressing coordinate v for the bottom right hand corner of the two dimensional rectangular object is set to be equal to the texture addressing offset in the y direction plus the texture addressing extent. Following along the illustration given above, the texture addressing coordinate u for the bottom right hand corner of the two dimensional rectangular object is 0.125 plus 0.333, which is equal to 0.458. The texture addressing coordinate v for the bottom right hand corner of the two dimensional rectangular object is 0.143 plus 0.333, which is equal to 0.476. Thus, the texture addressing coordinates (u, v) for the bottom right hand corner of the two dimensional rectangular object is (0.458, 0.476). The texture addressing coordinates (u, v) for each corner of the two dimensional rectangular object are illustrated in Figure 5.

[0037] In this manner, the coordinates for the vertices positioned on the upper left hand corner, the upper right hand corner, the bottom right hand corner and the bottom left hand corner of the two dimensional rectangular object are determined. The rest of the vertices on the two dimensional rectangular object may be determined by interpolating the texture addressing coordinates (u, v) and the (x, y) coordinates of the vertices on the upper left hand corner, the upper right hand corner, the bottom right hand corner and the bottom left hand corner of the two dimensional rectangular object. At the end of step 260, a two dimensional rectangular object is created with vertices that correspond with an area of the desktop display surface texture that will be mapped to the two dimensional rectangular object. The coordinates of the vertices are computed as a function of the zoom factor and the offsets received at step 250.

[0038] At step 270, the two dimensional rectangular object is rendered by mapping the desktop display surface texture (from step 240) to the two dimensional rectangular object, thereby creating a rendered two dimensional rectangular object. The two dimensional rectangular object may be rendered using an API, such as DirectX command, OpenGL and the like. At step 280, the rendered two dimensional rectangular object is stored in the render target surface 125. Following along the illustration given above, the rendered two dimensional rectangular object 625 is

illustrated in Figure 6. Notably, the desktop display surface 326 (shown in Figure 3) is zoomed or scaled up according to the zoom factor of three, an offset in the x direction of four pixels and an offset in the y direction of three pixels.

[0039] At step 290, the scanout read location is set to read from the render target surface 125. The scanout read location may be set by the graphics driver 134 in response to receiving commands from the scalable desktop software 132. The scanout control logic 120 then reads the rendered two dimensional rectangular object from the render target surface 125 and transmits the rendered two dimensional rectangular object to the display device 110 for display.

[0040] At step 295, a determination is made as to whether a new zoom factor or offsets have been received. If the answer is in the affirmative, then processing returns to step 260 at which another two dimensional rectangular object is created with a new set of vertices according to the new zoom factor and/or offsets. However, if the answer is in the negative, then processing returns to step 270 at which the same two dimensional rectangular object is rendered again.

[0041] While the foregoing is directed to embodiments of the present invention, other and further embodiments of the invention may be devised without departing from the basic scope thereof, and the scope thereof is determined by the claims that follow.